Improving random number generators by chaotic iterations. Application in data hiding

Christophe Guyeux, Qianxue Wang, Jacques Bahi

Improving random number generators by chaotic iterations Application in data hiding

Christophe Guyeux, Qianxue Wang and Jacques M. Bahi
University of Franche-Comte
Computer Science Laboratory LIFC, Besançon, France
Email: christophe.guyeux@univ-fcomte.fr, qianxue.wang@univ-fcomte.fr, jacques.bahi@univ-fcomte.fr

***Abstract*—In this paper, a new pseudo-random number generator (PRNG) based on chaotic iterations is proposed. This method also combines the digits of two XORshifts PRNGs. The statistical properties of this new generator are improved: the generated sequences can pass all the DieHARD statistical test suite. In addition, this generator behaves chaotically, as defined by Devaney. This makes our generator suitable for cryptographic applications. An illustration in the field of data hiding is presented and the robustness of the obtained data hiding algorithm against attacks is evaluated.**

***Keywords*—Topological chaos; Pseudo-random number generator; Statistical tests; Internet security; data hiding; Discrete chaotic iterations.**

I. Introduction

A pseudo-random number generator (PRNG) is an algorithm for generating a sequence of numbers that is supposed to be indistinguishable from a uniformly chosen random sequence [7]. The sequence is not truly random in that it is completely determined by a relatively small set of initial values, called the PRNG's seed. Compare to hardware-based approaches, these PRNGs must be easy to generate and process, but are less closer to truly random behavior. PRNGs play an important role in practice for a whole range of applications such as information security, statistics (samplings, simulations and Monte-Carlo techniques), video games and gambling machines, to name a few [4]. These PRNGs are often based on logical operations like bitwise exclusive or (XOR) and on circular shift of bit vectors. XORshift, designed by George Marsaglia [9], is a popular example of such generators. However, the security level of some PRNGs of this kind has been revealed to be inadequate by today's standards. We investigated whether it would be possible to combine two generators in some way that would give better properties than the individual components alone.

This paper extends the study started in [2] and [15]. In [2], it is proven that chaotic iterations (CIs), a suitable tool for fast computing iterative algorithms, satisfies the chaos property, as it is defined by Devaney [3]. In [15], the chaotic behavior of CIs are exploited in order to obtain an unpredictable PRNG. This generator, based on chaotic iterations, depends on two input sequences. In [15], these two sequences are constituted by two logistic maps. This novel generator has successfully passed the NIST (National Institute of Standards and Technology of the U.S. Government) battery of tests [10]. In this new paper, we achieve to improve the speed of the former PRNG, by using two XORshifts in place of the logistic map. In addition, this new version of our PRNG is able to pass the famous DieHARD statistical battery of tests [8]. And its security is improved compared to XORshift alone, and to our former PRNG. After presenting the theoretical framework of the study, a concrete example of how to use these pseudo-random numbers in the field of data hiding is detailed. An analysis focuses on the watermarked images which have already been subjected to common image distortion attacks. It is shown that sequences generated from this generator have a good robustness in the presence of such attacks.

The rest of this paper is organized in the following way: in Section II, some basic definitions concerning chaotic iterations and PRNGs are recalled. Then, the generator based on discrete chaotic iterations is presented in Section III. In Section IV, we show that the proposed PRNG passes the DieHARD statistical tests. In Sections V and VI, there is a discussion on a potential application scenario to watermarking. Finally, some conclusions and future work are drawn in Section VII.

II. Basic recalls

This section is devoted to basic notations and terminologies in the fields of chaotic iterations and PRNGs.

A. Notations

$[\![1;\mathsf{N}]\!] \to \{1,2,\ldots,N\}$
$s^n \to$ the n^{th} term of a sequence $s=(s^1,s^2,\ldots)$
$v_i \to$ the i^{th} component of an array $v=(v_1,v_2,\ldots)$
$f^k \to k^{th}$ composition of a function f

$$f^k = \underbrace{f\circ\ldots\circ f}_{k \text{ times}}$$

strategy $\to$ a sequence which elements belong in $[\![1;\mathsf{N}]\!]$
$\mathbb{S} \to$ the set of all strategies
$\oplus \to$ bitwise exclusive or
$+ \to$ the integer addition
$\ll$ and $\gg \to$ the usual shift operators

B. Chaotic iterations

Definition 1 The set $\mathbb{B}$ denoting $\{0,1\}$, let $f:\mathbb{B}^{\mathsf{N}}\longrightarrow\mathbb{B}^{\mathsf{N}}$ be an "iteration" function and $S\in\mathbb{S}$ be a chaotic strategy. Then, the so-called *chaotic iterations* are defined by

$$x^0\in\mathbb{B}^{\mathsf{N}},$$
$$\forall n\in\mathbb{N}^*, \forall i\in[\![1;\mathsf{N}]\!], x_i^n=\begin{cases} x_i^{n-1} & \text{if } S^n\neq i \\ f(x^{n-1})_{S^n} & \text{if } S^n=i. \end{cases} \quad (1)$$

In other words, at the n^{th} iteration, only the S^n–th cell is "iterated". Note that in a more general formulation, S^n can be a subset of components and $f(x^{n-1})_{S^n}$ can be replaced by

$f(x^k)_{S^n}$ (where $k < n$), thus describing for example, delays due to transmissions (see *e.g.* [1]). For the general definition of such chaotic iterations, see, e.g. [11].
Chaotic iterations generate a set of vectors (boolean vector in this paper), they are defined by an initial state x^0, an iteration function f and a chaotic strategy S.

C. XORshift

XORshift is a category of very fast PRNGs designed by George Marsaglia [9]. It repeatedly uses the exclusive or (XOR) on a number, with a bit shifted copy of itself by a positions either to the right or to the left, where $0 < a < w$ and $w = 32$ or 64. The initial state of a XORshift generator is a given vector of bits. At each step, the next state is obtained by applying a given number of XORshift operations as defined in Algorithm 1. This algorithm has a period of $2^{32}-1 = 4.29\times10^9$.

Input: x (a 32-bit word)
Output: r (a 32-bit word)
$x \leftarrow x \oplus (x \ll 13)$;
$x \leftarrow x \oplus (x \gg 17)$;
$x \leftarrow x \oplus (x \ll 5)$;
$r \leftarrow x$;
return r;

Algorithm 1: An arbitrary round of XORshift

D. Input sequences

In [15], we have used two logistic maps [14] as input sequences to define a novel PRNG (called CI PRNG) based on chaotic iterations. We have mathematically proven that it behaves chaotically, as defined by Devaney. In addition, this generator can successfully pass the NIST tests suite. However, chaotic systems like logistic maps work in the real numbers domain, and therefore a transformation from real numbers into integers is needed. This process leads to a degradation of the chaotic behavior of the generator and a lot of time wasted during computations [5]. Our purpose is then to improve the speed of this former generator and grant its chaotic properties by using a faster PRNG, namely XORshift. Moreover, we will show in Section IV-B that this new generator can pass the famous DieHARD battery of tests.

III. The generation of CI pseudo-random sequence

The design of the PRNG based on discrete chaotic iterations is proposed in this section, while its performance is evaluated in the next one.

A. Chaotic iterations as PRNG

The novel generator is designed by the following process. Let $\mathsf{N} \in \mathbb{N}^*, \mathsf{N} \geqslant 2$. Some chaotic iterations are done, which generate a sequence $(x^n)_{n\in\mathbb{N}} \in \left(\mathbb{B}^{\mathsf{N}}\right)^{\mathbb{N}}$ of boolean vectors: the successive states of the iterated system. Some of those vectors are chaotically extracted and their components constitute our pseudo-random bit flow. Chaotic iterations are realized as follows: initial state $x^0 \in \mathbb{B}^{\mathsf{N}}$ is a boolean vector taken as a seed and chaotic strategy $(S^n)_{n\in\mathbb{N}} \in [\![1, \mathsf{N}]\!]^{\mathbb{N}}$ is constructed with XORshift. Lastly, iterate function f is the vectorial boolean negation

$$f_0 : (x_1, ..., x_{\mathsf{N}}) \in \mathbb{B}^{\mathsf{N}} \longmapsto (\overline{x_1}, ..., \overline{x_{\mathsf{N}}}) \in \mathbb{B}^{\mathsf{N}}.$$

To sum up, at each iteration, only S^i-th component of state x^n is updated as follows

$$x_i^n = \begin{cases} x_i^{n-1} & \text{if } i \neq S^i, \\ \overline{x_i^{n-1}} & \text{if } i = S^i. \end{cases} \quad (2)$$

Finally, let $\mathcal{M}$ be a finite subset of $\mathbb{N}^*$. Some x^n are selected by a sequence m^n as the pseudo-random bit sequence of our generator. The sequence $(m^n)_{n\in\mathbb{N}} \in \mathcal{M}^{\mathbb{N}}$ is computed with XORshift. So, the generator returns the following values:

- the components of x^{m^0},
- following by the components of $x^{m^0+m^1}$,
- following by the components of $x^{m^0+m^1+m^2}$,
- *etc.*

In other words, the generator returns the following bits:

$$x_1^{m_0} x_2^{m_0} x_3^{m_0} \dots x_{\mathsf{N}}^{m_0} x_1^{m_0+m_1} x_2^{m_0+m_1} \dots x_{\mathsf{N}}^{m_0+m_1} x_1^{m_0+m_1+m_2} x_2^{m_0+m_1+m_2} \dots$$

and its k^{th} bit is equal to

$$x_{k+1 \pmod{\mathsf{N}}}^{\sum_{i=0}^{\lfloor k/\mathsf{N} \rfloor} m_i}.$$

The basic design procedure of the novel generator is summed up in Algorithm 2. The internal state is x, the output array is r; a and b are those computed by the two XORshift generators. Lastly, c and N are constants and $\mathcal{M}$ = {c,c+1} ($c \geqslant 3\mathsf{N}$ is recommended).

Input: the internal state x
(x is an array of N 1-bit words)
Output: an array r of N 1-bit words
$a \leftarrow XORshift1()$;
$m \leftarrow a \ mod \ 2 + c$;
for $i = 0, \dots, m$ **do**
  $b \leftarrow XORshift2()$;
  $S \leftarrow b \ mod \ \mathsf{N}$;
  $x_S \leftarrow \overline{x_S}$;
end
$r \leftarrow x$;
return r;

Algorithm 2: An arbitrary round of CI generator

B. Example

In this example, $\mathsf{N} = 5$ and $\mathcal{M}$ = {4,5} are adopted for easy understanding. The initial state of the system x^0 can be seeded by the decimal part of the current time. For example, the current time in seconds since the Epoch is 1237632934.484084, so $t = 484084$. $x^0 = t \pmod{32}$ in binary digits, then $x^0 = (1, 0, 1, 0, 0)$. m and S can now be computed from two XORshift PRNGs:

m = 4, 5, 4, 4, 4, 4, 5, 5, 5, 5, 4, 5, 4,...
S = 2, 4, 2, 2, 5, 1, 1, 5, 5, 3, 2, 3, 3,...

Chaotic iterations are made with initial state x^0, vectorial logical negation f_0 and strategy S. The result is presented in Table I. Let us recall that sequence m gives the states x^n to return: $x^4, x^{4+5}, x^{4+5+4}, \dots$
So, in this example, the output of the generator is: 10100111101111110011...

TABLE I
APPLICATION EXAMPLE

m :	4					5						4				
S	2	4	2	2		5	1	1	5	5		3	2	3	3	
x^0					x^4						x^9					x^{13}
1					1		$\overset{1}{\rightarrow}0$	$\overset{1}{\rightarrow}1$			1					1
0	$\overset{2}{\rightarrow}1$		$\overset{2}{\rightarrow}0$	$\overset{2}{\rightarrow}1$	1						1		$\overset{2}{\rightarrow}0$			0
1					1						1	$\overset{3}{\rightarrow}0$		$\overset{3}{\rightarrow}1$	$\overset{3}{\rightarrow}0$	0
0		$\overset{4}{\rightarrow}1$			1						1					1
0					0	$\overset{5}{\rightarrow}1$			$\overset{5}{\rightarrow}0$	$\overset{5}{\rightarrow}1$	1					1

Output: $x_1^0 x_2^0 x_3^0 x_4^0 x_5^0 x_1^4 x_2^4 x_3^4 x_4^4 x_5^4 x_1^9 x_2^9 x_3^9 x_4^9 \; x_5^9 x_1^{13} x_2^{13} x_3^{13} x_4^{13} x_5^{13}$... = 10100111101111110011...

C. Chaotic properties

Despite a large number of papers published in the field of chaos-based PRNGs, the impact that this research has made on conventional information security is rather marginal. This is due to the following reasons: almost all chaotic algorithms are based on dynamical systems defined on the set of real numbers. So these generators are usually slow, require considerably more storage space and lose some of their chaotic properties during computations. These major problems restrict their use in security fields as cryptography [6].

The PRNG proposed in this paper does not inherit its chaotic properties from a real chaotic map, but from chaotic iterations defined in Section II-B. It has been proven in [2] that CIs behave as chaos, as it is defined by Devaney: they are regular, transitive and sensitive to initial conditions. This most famous definition of chaotic behavior for a dynamical system implies various desired properties in information security, such as: unpredictability, mixture, sensitivity, and uniform repartition. The principal interest of CIs is that they can be used without real numbers. Indeed, the sequence inputed in chaotic iterations constitutes a coordinate of its initial state, and the chaotic behavior of a dynamical system does not depend on this initial state. So if we take integer sequences as input instead, then CIs become faster while preserving their chaotic properties. This allows the conception of a new generation of fast and chaotic PRNGs

IV. TESTING A GENERATOR

Here, the empirical tests have been carried out, making use of the DieHARD statistical test suite. In this section we will briefly review the approach taken together with key results and conclusions. It is not our intention to document these tests in detail in the present section, since it has been done several times in many other papers [8],[12],[13].

A. DieHARD battery of tests

DieHARD battery of tests has been a stringent standard for evaluating PRNGs for over a decade. Passing this battery is considered as a good rule of thumb to validate a PRNG. DieHARD battery consists of 18 different independent statistical tests. This collection of tests is based on assessing the randomness of bits comprising 32-bit integers obtained from a random number generator. Each test requires 2^{23} 32-bit integers in order to run the full set of tests.

Most of the tests in DieHARD return a $p-$value, which should be uniform on $[0, 1)$ if the input file contains truly independent random bits. Those $p-$values are obtained by $p = F(X)$, where F is the assumed distribution of the sample random variable $X-$often supposed as normal distribution. But that assumed F is just an asymptotic approximation, for which the fit will be worst in the tails. Thus occasional $p-$values near 0 or 1, such as 0.0012 or 0.9983, are not surprising. An individual test is considered to be a failure if its $p-$value approaches 1 more closely, for example $p > 0.9999$.

B. Analysis

Table II gives the results derived from applying the DieHARD battery of tests to the RNGs considered in this work. As it can be observed, the results of the individual tests Count the ones 1, Binary Rank 31×31 and Binary Rank 32×32 show that in the random numbers obtained with the XORshift generator only the least significant bits seem to be independent. This explains the poor behavior of this RNG in the aforementioned basic tests that evaluate the independence of real numbers. But the generator based on discrete chaotic iterations can pass all the DieHARD battery of tests. This proves that the security of the given generator has been improved by CIs.

V. APPLICATION EXAMPLE IN DIGITAL WATERMARKING

Information hiding has recently become a major information security technology, especially with the increasing importance and widespread distribution of digital media through the Internet [16]. It includes several techniques like digital watermarking. The aim of digital watermarking is to embed a piece of information into digital documents, such as pictures or movies. This is for a large panel of reasons, such as: copyright protection, control utilization, data description, content authentication, and data integrity. For these reasons, many different watermarking schemes have been proposed in recent years. Digital watermarking must have essential characteristics, including: security, imperceptibility, and robustness. Chaotic methods have been proposed to encrypt the watermark before embedding it in the carrier image for these security reasons. In this paper, a new watermarking algorithm is given. It is based on the chaotic PRNG presented above.

A. Most and least significant coefficients

Let us first introduce the definitions of most and least significant coefficients.

TABLE II
RESULTS OF DIEHARD BATTERY OF TESTS

No.	Test name	Generators	
		XORshift	PRNG (Chaotic iterations)
1	Overlapping Sum	Pass	Pass
2	Runs Up 1	Pass	Pass
	Runs Down 1	Pass	Pass
	Runs Up 2	Pass	Pass
	Runs Down 2	Pass	Pass
3	3D Spheres	Pass	Pass
4	Parking Lot	Pass	Pass
5	Birthday Spacing	Pass	Pass
6	Count the ones 1	Fail	Pass
7	Binary Rank 6×8	Pass	Pass
8	Binary Rank 31×31	Fail	Pass
9	Binary Rank 32×32	Fail	Pass
10	Count the ones 2	Pass	Pass
11	Bit Stream	Pass	Pass
12	Craps Wins	Pass	Pass
	Throws	Pass	Pass
13	Minimum Distance	Pass	Pass
14	Overlapping Perm.	Pass	Pass
15	Squeeze	Pass	Pass
16	OPSO	Pass	Pass
17	OQSO	Pass	Pass
18	DNA	Pass	Pass
	Number of tests passed	15	18

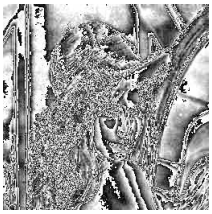
Definition 2 For a given image, the most significant coefficients (in short MSCs), are coefficients that allow the description of the relevant part of the image, *i.e.* its most rich part (in terms of embedding information), through a sequence of bits.

For example, in a spatial description of a grayscale image, a definition of MSCs can be the sequence constituted by the first three bits of each pixel as shown in Fig. 1(a). In a discrete cosine frequency domain description, each 8×8 block of the carrier image is mapped to a list of 64 coefficients. The energy of the image is contained in the first of them. After binary conversion, the first fourth coefficients of all these blocks can constitute a possible sequence of MSCs.

Definition 3 By least significant coefficients (LSCs), we mean a translation of some insignificant parts of a medium in a sequence of bits (insignificant can be understand as: "which can be altered without sensitive damages").

These LSCs can be for example, the last three bits of the gray level of each pixel, in the case of a spatial domain watermarking of a grayscale image, as in Fig. 1(b). Discrete cosine, Fourier, and wavelet transform can be used to define LSCs and MSCs, in the case of frequency domain watermarking, among other possible choices. Moreover, these definitions are not limited to image media, but can easily be extended to the audio and video media as well.

(a) MSCs of Lena



(b) LSCs of Lena

Fig. 1. Spatial MSCs and LSCs of Lena.

LSCs are used during the embedding stage: some of the least significant coefficients of the carrier image will be chaotically chosen and replaced by the bits of the mixed watermark. With a large number of LSCs, the watermark can be inserted more than once and thus the embedding will be more secure and robust, but also more detectable.

The MSCs are only useful in the case of authentication: encryption and embedding stages depend on them. Hence, a coefficient should not be defined at the same time, as a MSC and a LSC; the last can be altered, while the first is needed to extract the watermark.

B. Stages of the algorithm

Our watermarking scheme consists of two stages: (1) mixture of the watermark and (2) its embedding.

1) Watermark mixture.: Firstly, for safety reasons, the watermark can be mixed before its embedding into the image. A common way to achieve this stage is to use the bitwise exclusive or (XOR), for example, between the watermark and the above PRNG. In this paper, we will use another mixture scheme based on chaotic iterations. Its chaotic strategy, defined with our PRNG, will be highly sensitive to the MSCs, in the case of an authenticated watermark, as stated in [2].

2) Watermark embedding.: Some LSCs will be substituted by all bits of the possibly mixed watermark. To choose the sequence of LSCs to be altered, a number of integers, less than or equal to the number N of LSCs corresponding to a chaotic sequence $\left(U^k\right)_k$, is generated from the chaotic strategy used in the mixture stage. Thus, the U^k-th least significant coefficient of the carrier image is substituted by the k^{th} bit of the possibly mixed watermark. In the case of authentication, such a procedure leads to a choice of the LSCs which are highly dependent on the MSCs. For the detail of this stage see Section VI-A2.

3) Extraction.: The chaotic strategy can be regenerated, even in the case of an authenticated watermarking because the MSCs have not been changed during the stage of embedding the watermark. Thus, the few altered LSCs can be found, the mixed watermark can then be rebuilt, and the original watermark can be obtained. If the watermarked image is attacked, then the MSCs will change. Consequently, in the case of authentication and due to the high sensitivity of the embedding sequence, the LSCs designed to receive the watermark will be completely different. Hence, the result of the recovery will have no similarity with the original watermark: authentication is reached.

VI. EVALUATION OF ROBUSTNESS

In this section, a complete application example of the above chaotic watermarking method is given and its robustness to some attacks is studied. This case study enables us to precise the details of the algorithm and evaluate it.

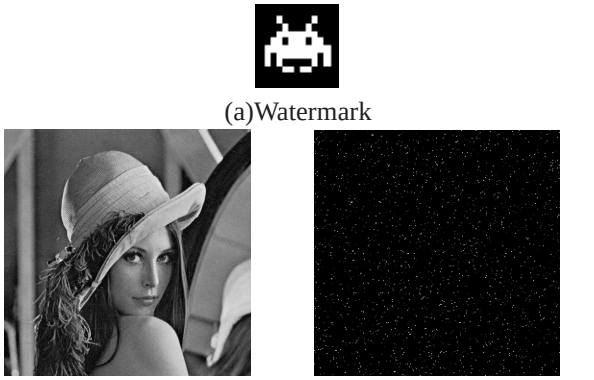
A. Stages and details

1) Images description.: Carrier image is Lena, a 256 grayscale image of size 256×256. The watermark is the 64×64

TABLE III
CROPPING ATTACKS

UNAUTHENTICATION		AUTHENTICATION	
Size (pixels)	Similarity	Size (pixels)	Similarity
10	99.48%	10	49.68%
50	97.63%	50	54.54%
100	91.31%	100	52.24%
200	68.56%	200	51.87%

pixels binary image depicted in Fig. 2(a). The embedding domain will be the spatial domain. The selected MSCs are the four most significant bits of each pixel and the LSCs are the three last bits (a given pixel will at most be modified of four levels of gray by an iteration). Before its embedment, the watermark is mixed with chaotic iterations. The system to iterate, chaotic strategy S^n and iterate function are defined below.



(a)Watermark

(b)Watermarked Lena (c)Differences with original.

Fig. 2. Watermarked Lena and differences

2) Embedding of the watermark.: To embed the watermark, the sequence $(U^k)_{k\in\mathbb{N}}$ of altered bits taken from the M LSCs must be defined. To do so, the strategy $(S^k)_{k\in\mathbb{N}}$ of the encryption stage is used as follows

$$\begin{cases} U^0 &= S^0 \\ U^{n+1} &= S^{n+1} + 2 \times U^n + n \ (mod\ \mathsf{M}) \end{cases} \quad (3)$$

to obtain the result depicted in Fig. 2(b). The map $\theta \mapsto 2\theta$ of the torus, which is a famous example of topological Devaney's chaos [3], has been chosen to make $(U^k)_{k\in\mathbb{N}}$ highly sensitive to the chaotic strategy $(S^k)_{k\in\mathbb{N}}$. As a consequence, $(U^k)_{k\in\mathbb{N}}$ is highly sensitive to the alteration of the MSCs. In case of authentication, any significant modification of the watermarked image will lead to a completely different extracted watermark.

B. Simulation results

To prove the efficiency and the robustness of the proposed algorithm, some attacks are applied to our chaotically watermarked image. For each attack, a similarity percentage with the original watermark is computed. This percentage is the number of equal bits between the original and the extracted watermark, shown as a percentage. A result less than or equal to 50% implies that the image has probably not been watermarked.

1) Cropping attack: In this kind of attack, a watermarked image is cropped. In this case, the results in Table III have been obtained.

TABLE IV
ROTATION ATTACKS

UNAUTHENTICATION		AUTHENTICATION	
Angle (degree)	Similarity	Angle (degree)	Similarity
2	97.41%	2	70.01%
5	94.67%	5	59.47%
10	91.30%	10	54.51%
25	80.85%	25	50.21%

TABLE V
JPEG COMPRESSION ATTACKS

UNAUTHENTICATION		AUTHENTICATION	
Compression	Similarity	Compression	Similarity
2	82.95%	2	54.39%
5	65.23%	5	53.46%
10	60.22%	10	50.14%
20	53.17%	20	48.80%

In Fig. 3, the decrypted watermarks are shown after a crop of 50 pixels and after a crop of 10 pixels, in the authentication case.

(a) Unauthentication (10×10).

(b) Authentication (10×10). (c) Unauthentication (50×50).

Fig. 3. Extracted watermark after a cropping attack.

By analyzing the similarity percentage between the original and the extracted watermark, we can conclude that in the case of unauthentication, the watermark still remains after a cropping attack. The desired robustness is reached. It can be noticed that cropping sizes and percentages are rather proportional. In the case of authentication, even a small change of the carrier image (a crop by 10×10 pixels) leads to a really different extracted watermark. In this case, any attempt to alter the carrier image will be signaled, thus the image is well authenticated.

2) Rotation attack: Let r_θ be the rotation of angle θ around the center $(128, 128)$ of the carrier image. So, the transformation $r_{-\theta} \circ r_\theta$ is applied to the watermarked image. The results in Table IV have been obtained. The same conclusion as above can be declaimed.

3) JPEG compression: A JPEG compression is applied to the watermarked image, depending on a compression level. This attack leads to a change of the representation domain (from spatial to DCT domain). In this case, the results in Table V have been obtained, illustrating a good authentication through JPEG attack. As for the unauthentication case, the watermark still remains after a compression level equal to 10. This is a good result if we take into account the fact that we

TABLE VI
GAUSSIAN NOISE ATTACKS

UNAUTHENTICATION		AUTHENTICATION	
Standard dev.	Similarity	Standard dev.	Similarity
1	74.26%	1	52.05%
2	63.33%	2	50.95%
3	57.44%	3	49.65%

use spatial embedding.

4) Gaussian noise: A watermarked image can be also attacked by the addition of a Gaussian noise, depending on a standard deviation. In this case, the results in Table VI are obtained.

VII. CONCLUSIONS AND FUTURE WORK

In this paper, the PRNG proposed in [15] is improved, by using the famous XORshift generator. By combining these components with chaotic iterations, we define a faster generator with chaotic properties. In addition to achieving the NIST tests suite, this new generator successfully passes all the stringent DieHARD battery of tests. The randomness and disorder generated by this algorithm has been evaluated. It offers a sufficient level of security for a whole range of computer usages. An application example in the field of data hiding is given and its robustness through attacks is studied. In future work, the speed of our generator will be improved again, the comparison of different chaotic strategies will be explored, and other iteration functions will be studied. Finally, new applications in computer science security field will be proposed.

REFERENCES

[1] J. M. Bahi. Boolean totally asynchronous iterations. *Int. Journal of Mathematical Algorithms*, 1:331–346, 2000.

[2] J. M. Bahi and C. Guyeux. Topological chaos and chaotic iterations, application to hash functions. *WCCI'10: 2010 IEEE World Congress on Computational Intelligence*, Accepted paper, 2010.

[3] R. L. Devaney. *An Introduction to Chaotic Dynamical Systems*. Redwood City: Addison-Wesley, 2nd edition, 1989.

[4] C.M. Gonzaleza, H.A. Larrondo, and O.A. Rosso. Statistical complexity measure of pseudorandom bit generators. *Physica A: Statistical Mechanics and its Applications*, 354:281–300, 2005.

[5] A. Kanso and N. Smaoui. Logistic chaotic maps for binary numbers generations. *Chaos, Solitons and Fractals*, 40:2557–2568, 2009.

[6] L. Kocarev. Chaos-based cryptography: a brief overview. *IEEE Circ Syst Mag*, 7:6–21, 2001.

[7] P. L'ecuyer, DIRO, CIRRELT, and GERAD. Comparison of point sets and sequences for quasi-monte carlo and for random number generation. *SETA 2008*, LNCS 5203:1–17, 2008.

[8] G. Marsaglia. Diehard: a battery of tests of randomness. *http://stat.fsu.edu/ geo/diehard.html*, 1996.

[9] G. Marsaglia. Xorshift rngs. *Journal of Statistical Software*, 8(14):1–6, 2003.

[10] NIST Special Publication 800-22 rev. 1. A statistical test suite for random and pseudorandom number generators for cryptographic applications. August 2008.

[11] F. Robert. *Discrete Iterations. A Metric Study*, volume 6. Springer Series in Computational Mathematics, 1986.

[12] S. K. Tan and S.U. Guan. Randomness quality of permuted pseudo-random binary sequences. *Mathematics and Computers in Simulation*, 79:1618–1626, 2009.

[13] M. S. Turan, A Doganaksoy, and S Boztas. On independence and sensitivity of statistical randomness tests. *SETA 2008*, LNCS 5203:18–29, 2008.

[14] S. M. Ulam and J. V. Neumann. On combination of stochastic and deterministic processes. *Amer. Math. Soc.*, 53:1120, 1947.

[15] Q Wang, C Guyeux, and J M. Bahi. A novel pseudo-random generator based on discrete chaotic iterations for cryptographic applications. In *First International Conference on Evolving Internet*, 2009.

[16] X. Wu and Z. Guan. A novel digital watermark algorithm based on chaotic maps. *Physical Letters A*, 365:403–406, 2007.